iRMX 80™ INTERACTIVE CONFIGURATION UTILITY USER'S GUIDE

Manual Order Number: 142603-002

A243/880/2.5K DD

PREFACE

This manual describes the use of the iRMX 80 Interactive Configuration Utility (ICU80), which is provided as an alternate and simplified means of configuring iRMX 80 application systems. Because you configure a system after programming it, you should become familiar with the iRMX 80 User's Guide (manual order number 9800522) before studying this manual.

The following manuals provide information related to the iRMX 80 Real-Time, Multitasking Executive:

- iRMX 80 Installation Instructions, 9803087.
- iRMX 80 User's Guide, 9800522.
- ISIS-II User's Guide, 9800306.
- 8080/8085 Assembly Language Reference Manual, 9800301.
- PL/M Programming Manual, 9800268.
- ISIS-II PL/M Compiler Operator's Manual, 9800300.
- BASIC-80 Reference Manual, 9800758.
- FORTRAN-80 Programming Manual, 9800481.
- ISIS-II FORTRAN-80 Compiler Operator's Manual, 9800480

CONTENTS

TABLES

ILLUSTRATIONS

CHAPTER 1
INTRODUCTION

An iRMX 80 application system consists of software from two sources: general-purpose iRMX 80 software, provided by Intel; and the software you have written to solve a particular problem. After you have written your application software, you must combine it with iRMX 80 software. The act of combining these software components is called software configuration or, more simply, configuration. In addition to linking together these two kinds of software, the configuration process involves selecting those iRMX 80 modules actually needed in your application.

The Nucleus of iRMX 80, which is included in every system and which performs a control function, must be provided with complete information concerning the hardware and software environments of your system. Consequently, configuring requires you to describe formally both your hardware and your software; it is this fact that can make configuration a tedious and somewhat difficult process.

Your iRMX 80 package provides you with two avenues for configuring applications. One way is manually to build configuration tables. This method is exacting and requires you to have some knowledge of the internals of iRMX 80. However, it is occasionally necessary for experienced users who have done such advanced things as writing their own internal structures. The other way to configure is to use the Interactive Configuration Utility (ICU80), which is on your iRMX 80 Interactive Configuration Utility diskette.

ADVANTAGES OF CONFIGURING INTERACTIVELY

The Interactive Configuration Utility is designed to provide relief from the burden of manually creating configuration tables. It elicits the necessary information from you, at the terminal of your Intellec Development System, by asking simple questions about your application system. Configuring in this way is easier, faster, and less prone to errors of omission. Moreover, it requires you to have much less knowledge of the data structures that the iRMX 80 Nucleus uses to generate a working system.

USE ENVIRONMENT

The ICU80 utility runs on an Intellec Development System with the following adjuncts:

- Version 3.4 or later of ISIS-II.
- 64K bytes of RAM.
- Two single density floppy disk drives, or one of double density.
- An operator's terminal, unless one is included in your development system.

ICU80 CAPABILITIES

The ICU80 utility operates in two stages. The first stage consists of building a description file, which contains raw information about your system. Figure 1-1 illustrates the stage one process.

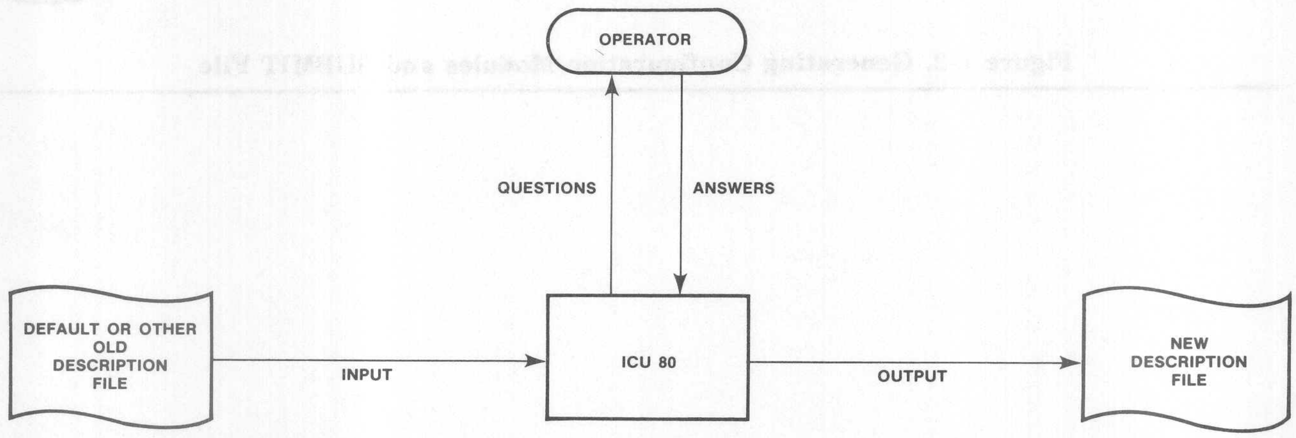

Figure 1-1. Editing a Description File

When you begin to use the ICU80 utility, the default description file, which is provided by ICU80, contains default answers to questions that will be asked at the terminal. During stage one, the answers that you provide replace, and in some cases add to, the default answers in the file. After completing the first stage process, you can make changes to your description file by inputting it again as an old description file. In this manner, a description file can evolve incrementally.

When you input an existing description file for editing, you must use the same version of the ICU80 utility under which the file was created. To ensure that you will not forget which version a description file was created under, you can put that information in a comment at the beginning of the file. Comments are discussed in the next chapter.

When your description file is developed to your satisfaction, you proceed to the second stage. The output from this stage consists of configuration modules and an ISIS-II SUBMIT file. Figure 1-2 illustrates the stage two process. The SUBMIT file, when executed, does the remaining housekeeping, including linking and locating all of the software for the application system. After executing the SUBMIT file, your application system is ready to be tested.

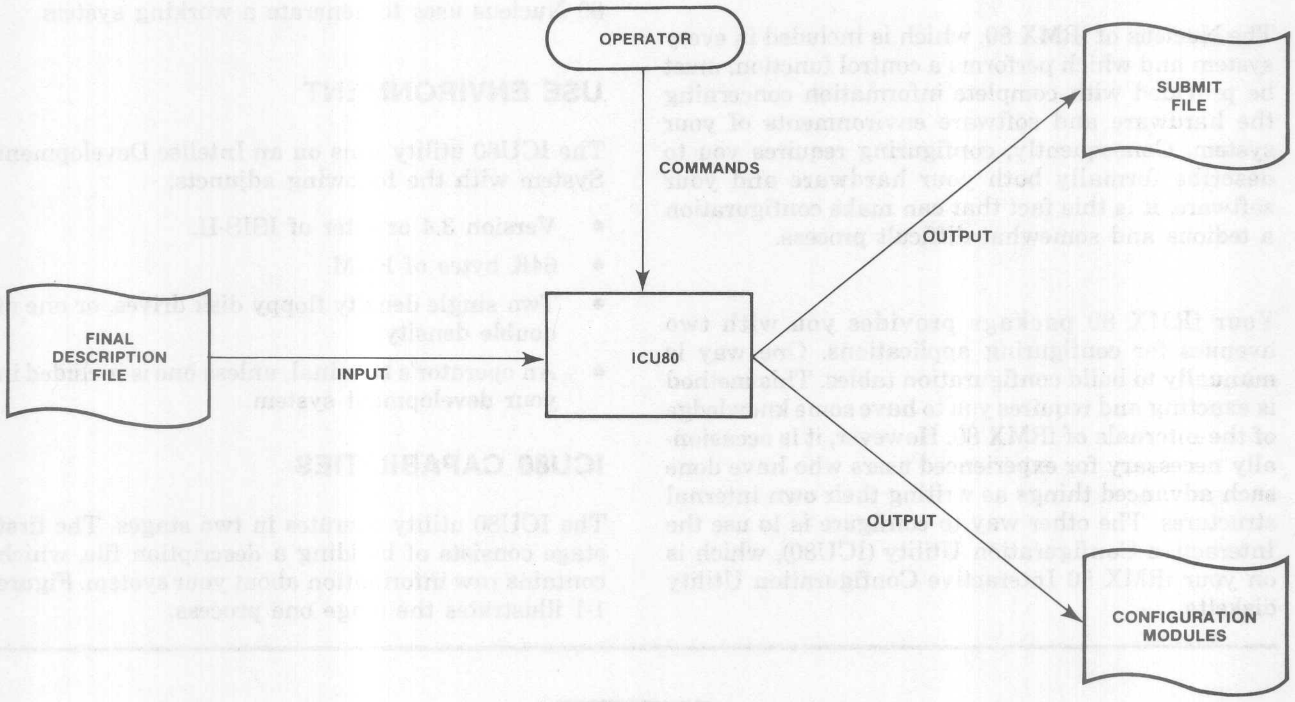

Figure 1-2. Generating Configuration Modules and SUBMIT File

CHAPTER 2
USING THE ICU80 UTILITY

The ICU80 utility can be used in either of two modes. The **command mode** allows you to perform functions other than editing, such as starting an edit session, listing the contents of the description file, or initiating the second stage of the configuration process. The **editing mode,** which is discussed later in this chapter, allows you to alter the contents of the description file.

INITIATING ICU80

The syntax for invoking the ICU80 utility is

ICU80 [input file name TO] output file name

where the elements inside the brackets are optional. The following cases apply:

- If only the output file name is included, and that file does not yet exist, then the system default description file is automatically used as input for editing. After editing, the edited file is stored in the named output file.
- If only the output file name is included, and that file already exists, then a backup (name "output file name.BAK") of that file is used as input for editing. After editing, the edited file is stored in the named output file.
- If both input and output files are included in the invocation, the input file must exist and is used as input for editing; the output file, on the other hand, must not exist. After editing, the edited file is stored in the named output file.

GETTING CLARIFICATION

At any time, while using the ICU80 utility, you can get information about the options available to you by entering a question mark (?), followed by a carriage return. For example, if you don't understand a prompt, you can use a question mark to get a description of the possible responses.

THE COMMAND MODE

When the ICU80 utility comes up, it displays a list of the commands available in the command mode, followed by a request that you enter a command. The commands are listed here in summary and then are defined in detail. Bear in mind that each command sequence must be terminated with a carriage return. Table 2-1 summarizes the commands available in the command mode.

Table 2-1. Commands in the Command Mode

Command	Meaning
C	Create a description file or change an entry in a description file.
L	List the contents of a description file.
E	Exit ICU80 with edited description file intact.
G	Generate the configuration modules and the SUBMIT file.
Q	Exit ICU80 with the description file unaffected by editing in the current ICU80 session.
R	Replace the current control character.

Complete descriptions of the commands are as follows:

C Allows you to begin editing the description file, either at the beginning or at a specific entry. The syntax of the C command is

C[hange][parameter[,value[,parameter]]]

where the elements inside the brackets are optional.

To create a new description file (complete with the standard default values), or to enter at the beginning of an existing description file, use C. When the ICU80 utility prompts with CONFIG: FILNAM.CNF, where FILNAM is the file name you supplied when invoking the ICU80 utility, you are ready to begin editing the description file.

The way you enter a file at a particular parameter depends on the type of parameter. (If you don't understand the remainder of this explanation, you should read it again after finishing this chapter.) If the parameter is of the single-response variety (like TERM HNDLR), enter "C", followed by the name of the parameter (like C TERM HNDLR). If the parameter is in a list (like TASK NAME: TSK1), enter "C", followed by the parameter and its current value (like C TASK NAME, TSK1). If the parameter is secondary (like PRIORITY, under the heading TASK NAME: TSK1), enter "C", followed by the main heading, its value, and the secondary parameter (like C TASK NAME, TSK1, PRIORITY).

It is not possible to enter a description file at a third-level parameter.

L	Lists the contents of the description file at the specified ISIS-II device/file. If no device/file name is given, the terminal (:CO:) is assumed. The syntax of the L command is: L[ist][device/file name] where the elements inside the brackets are optional.
E	Exits the ICU80 utility with the description file reflecting **all** of the changes from the current ICU80 session.
G	Generates, on command, the configuration modules and the SUBMIT file that are needed to complete the configuration process
Q	Exits the ICU80 utility with the description file reflecting **none** of the changes from the current ICU80 session.
R	Replaces the control character that is used in the special editing commands. The default character is the ESCape key, but you may replace it with any character of your choosing. To replace the current control character, enter R followed by a carriage return. Instructions as to what to do next will appear at your terminal. You may then enter the new control character.

DESCRIPTION FILES

An ICU80 description file contains information about both the hardware and the software in your system. Specific examples include the following:

- Which iRMX 80 components are needed in your system.
- Information about each of your initial tasks, such as user task name, priority, start address, stack size, etc.
- The names of each of your initial exchanges.
- A description of the peripheral hardware devices needed by your system.

Each description file contains information of two types: single responses and lists. An example of the single-response type of information is whether your system needs the Free Space Manager; either it does or it doesn't, so the information can always be provided by a single response. Lists, on the other hand, contain a variable number of entries; examples of lists include your tasks, your exchanges, and the disk controllers in your hardware configuration.

EDITING A DESCRIPTION FILE

The ways in which information is elicited and stored in a description file are different for the two types of information. Single-response parameters always have values. Initially the values are the defaults provided by the ICU80 utility. When editing, you are provided (at your terminal) with parameter names and the current values associated with those names. For example, on the first pass through a description file, one of the prompts is

```
TERM HNDLR: NONE
```

followed by a blinking cursor. If you want no terminal handler, you signify that the current value is satisfactory by entering a carriage return. In case you do want the terminal handler, you signify your choice by entering it before hitting carriage return. In the example, the alternatives to "NONE" are "MINIMAL" and "FULL".

If you do not know the alternatives to "NONE", you could obtain this information by entering "?" and a carriage return.

In contrast to single-response parameters, lists are initially empty. That is, a default list has no entries, and this fact is indicated by means of the end-of-list symbol, which is "***". For example, on the first pass through a description file, one of the prompts is

```
TASK NAME: ***
```

followed by a blinking cursor. Because you probably have at least one task, you would respond with a name for one of your tasks, after which the ICU80 utility would ask some questions about that task. When you have answered those questions, the ICU80 utility again prompts with

```
TASK NAME: ***
```

followed by a blinking cursor. This pattern is repeated until you have provided the ICU80 utility with data about all of your tasks. You signal that you have no more task names for the list by responding to the prompt for a task name with a carriage return.

CONSTANTS AND NAMES IN DESCRIPTION FILES

The ICU80 utility recognizes device or file names, identifiers, and several varieties of integer constants:

- A device/file name can be any device/file name acceptable to ISIS-II.
- An identifier can be any identifier that is compatible with the language you are using. For example, if you are using assembly language, identifiers can be up to six characters in length, counting dollar signs. On the other hand, if you are using PL/M, identifiers can be up to 32 characters in length, not counting dollar signs.

(If you use dollar signs, they are edited out immediately, so they will not appear when you list the description file..)

- Constants must be unsigned integers and can be entered in any of four formats. A format is specified by means of the presence or absence of a trailing symbol. Table 2-2 associates trailing symbols and allowable ranges with each of the formats.

Table 2-2. Integer Constant Formats

Format	Trailing Symbol	Range
Decimal	None or D	0-65535 (0D-65535D)
Hexadecimal	H	0H-0FFFFH
Octal	Q or O	0Q-177777Q (0O-177777O)
K-type*	K	0K-64K

* 1K = 1024 = 400H = 2000Q

COMMENTS IN DESCRIPTION FILES

A description file can contain comment lines to add clarity for the benefit of persons who might later be updating the file. A comment can be added by means of the (ESC) R command, and it can be deleted by the (ESC) D command. Both (ESC) R and (ESC) D are explained later.

SPECIAL COMMANDS FOR EDITING

To simplify the editing process, several special commands are available to you. They are listed here in summary fashion and then are defined in detail. Most of them are initiated by entering ESCape and then one or more other characters. We write (ESC) to indicate the ESCape character. Bear in mind that each command sequence must be terminated with a carriage return.

A brief summary of the special editing commands is given in Table 2-3.

Table 2-3. Special Editing Commands

Command Sequence	Effects
(ESC) B	Back up to previous line.
(ESC) C	Return to command mode.
(ESC) D	Delete current list entry or comment line.
(ESC) F parms	Find and display the designated parameter.
(ESC) H	Display this list of special commands.
(ESC) I	Insert list element ahead of current element.
(ESC) R	Insert comment ahead of current line.
?	Provide explanation.
(ESC) ?	Provide explanation.

Complete descriptions of the special editing commands are as follows:

(ESC) B Allows you to back up from the current prompt to the previous prompt. The previous prompt is displayed and you may continue as usual either by changing the current value or by entering carriage return. Backing up beyond the beginning of the description file returns you to the command mode.

(ESC) C Returns you to the command mode. The parameter currently being displayed is not changed.

(ESC) D Deletes the current list or comment from the description file. If the current entry is an indispensable secondary element in a list (such as a priority parameter in a list of tasks), then the entire list entry is deleted. If the entry currently being displayed is of the single-response type, (ESC) D has no effect on the entry.

(ESC) F parms Finds and displays the parameter indicated by the parameter string. The syntax of the (ESC) F command is

(ESC) F [parameter[, value[, parameter]]]

where the elements inside the brackets are optional. If you want to find a uniquely named parameter, like TERM HNDLR:, you need only enter (ESC) F TERM HNDLR. However, if you are seeking a parameter that is not unique because it is a secondary parameter in a list, you must enter enough information to specify the parameter. For example, suppose part of your description file looks like this:

```
TASK NAME: TSK1
  ENTRY POINT: TSK1
  PRIORITY: 35
  STK LENGTH: 24
  DFLT EXCHG: RQL2EX
TASK NAME: TSK2
  ENTRY POINT: TSK2
  PRIORITY: 83
  STK LENGTH: 40
  DFLT EXCHG: RQL5EX
TASK NAME: ***
```

If you want to change the priority of the task called TSK2, you can find that entry by entering (ESC) F TASK NAME, TSK2, PRIORITY.

The ICU80 utility responds by printing PRIORITY: 83, followed by the blinking cursor. At that time you can change the priority by entering the new value and hitting carriage return, If, instead, you want to place an additional task in the task list, you can find the end of the task list by entering (ESC) F TASK NAME ***.

(ESC) H Displays the list of special editing commands.

(ESC) I Allows you to insert an element ahead of the current list element, provided that the current element is a primary list element. Otherwise, (ESC) I has no effect. For example, suppose you want to insert a new task in the task list of the earlier example. If the insertion is to be between the first and second elements of the task list, find the second task name entry. That is, position the cursor so that your terminal shows

```
TASK NAME: TSK2
```

Now enter (ESC) I and carriage return, and ICU80 responds by printing

```
TASK NAME:
```

followed by the blinking cursor. You may now enter the information about the new task.

(ESC) R Allows you to insert a comment line immediately ahead of the current line. When you enter (ESC) R, a colon (:) is displayed on the screen, signalling that the remainder of that line is a comment. You now enter a comment of at most 120 characters and terminate the comment by entering a carriage return. Comments can be deleted by means of the (ESC) D command.

? Provides you with an explanation regarding the current prompt.

(ESC) ? Provides you with an explanation regarding the current prompt.

LINKING AND LOCATING THE APPLICATION SYSTEM

After the description file is developed, and the G command has been executed, you must SUBMIT the SUBMIT file that was generated by the G command. The name of the SUBMIT file is user-modifiable and is found in the second non-comment line of the description file. The syntax of the SUBMIT command is

```
SUBMIT device/file name
```

SUMMARY OF THE CONFIGURATION PROCESS

The following algorithm shows the main steps involved in configuration:

1. Plan your application, including defining tasks, exchanges, etc.
2. Write your code and compile the modules.
3. Activate the ICU80 utility and use the C command to begin developing your description file.
4. Finish developing your description file.
5. Use the G command to generate the configuration modules and the SUBMIT file.
6. SUBMIT the SUBMIT file to link the modules and locate your application system.
7. Test your application system.
8. If necessary, return to step 1, 2, or 3.

APPENDIX A ICU DISK FILES

FILE NAMES

Temporary:

The following files are used by ICU80 as work areas. They will be deleted by ICU80 when Exit or Quit is used to terminate ICU80. Note that if these files exist when ICU80 is initiated they will be destroyed.

ICU801.TMP

ICU802.TMP

ICU803.TMP

ICU804.TMP

Permanent Diskette Files:

These files are distributed with the iRMX 80 system and are located on the disk labeled INTERACTIVE CONFIGURATOR for iRMX 80. When using ICU80 all of these files must be located on the same diskette.

ICU80

ICU80.DDF

ICU80.OV1

ICU80.OV2

ICU80.OV3

ICU80.OV4

ICU80.OV5

ICU80.MCF

APPENDIX B
SAMPLE CONFIGURATION LISTING

```
: THIS IS AN EXAMPLE OF USING THE INTERACTIVE CONFIGURATION UTILITY
: TO CONFIGURE AN iRMX 80 APPLICATION SYSTEM.
: THE KEY CHARACTERISTICS OF THE APPLICATION ARE:
: iRMX 80 TASKS: I/O TERMINAL HANDLER, FREE SPACE MANAGER, AND
: ACTIVE DEBUGGER.
: THERE ARE THREE USER TASKS CALLED USER1, USER2, AND USER3. USER1 SERVICES
: INTERRUPTS FROM A HIGH SPEED PERIPHERAL DEVICE. USER2 AND USER3 ARE
: NOT INTERRUPT DRIVEN.
: EXPLANATIONS OF THE PARAMETER REQUIREMENTS CAN BE OBTAINED BY
: TYPING A QUESTION MARK (?) AFTER THE GIVEN PROMPT.
:
:
CONFIG:      :F1:EXAMPL.CNF
SUBMIT:      :F1:EXAMPL.CSD
:
: THE CAM MODULE IS NOT USED IF THE SYSTEM DOES NOT INCLUDE DFS.
:
CAM:      :F1:EXAMPL.CAM
LINKOBJ:       :F1:EXAMPL.LNK
LINKMAP:        :F1:EXAMPL.LEM
LOCOBJ:      SYSTEM.LOC
LOCMAP:       :F1:EXAMPL.LOM
CAMOBJ:       :F1:EXAMPL.LOC
CAMMAP:        :F1:EXAMPL.CLM
BOOT LDR:       NO
LOADABLE:        NO
CPU TYPE:       80/20
:
: TO INCLUDE A SYSTEM LEVEL TASK, IT IS NECESSARY ONLY TO
: RESPOND PROPERLY TO THE PROMPTS.
: ICU80 WILL AUTOMATICALLY INCLUDE THE NEEDED EXCHANGES AND TASKS
: AND MAKE THE CORRECT ENTRIES IN THE INITIAL TASK TABLE AND
: INITIAL EXCHANGE TABLES
:
TERM HNDLR:        FULL
  FUNCT:        I/O
  RATE:      1200
  CNTL TBL:        NO
  THISTK:       36
  THIPRI:       12
FSM:      ?
  THE FREE SPACE MANAGER PARAMETER CAN ASSUME ANY OF THE FOLLOWING VALUES:
  YES AND NO ARE THE ACCEPTABLE PARAMETER VALUES
  YES— THE TASKS AND EXCHANGES NEEDED TO INCLUDE THE FSM IN THE APPLICATION
  SYSTEM WILL BE AUTOMATICALLY INCLUDED.
  NO— THE FREE SPACE MANAGER WILL NOT BE INCLUDED
  TASK DESCRIPTOR ADDRESS = RQFSMD
FSM:       YES
  FSMSTK:       40
  FSMPRI:       130
DEBUG:      ACTIVE
  DBGSTK:       64
  DBGPRI:       140
ANLG HDLR:        NONE
:
```

```
: TO INCLUDE A USER TASK THE PROMPTS MUST BE ANSWERED. HOWEVER, IF
: THE TASK HAS NO DEFAULT EXCHANGE, NO VALUE SHOULD BE ENTERED FOR
: THE DEFAULT EXCHANGE.
: ICU80 WILL AUTOMATICALLY BUILD THE PROPER DATA STRUCTURE IN THE
: INITIAL TASK TABLE FOR THE USER TASK.
:
TASK NAME:        USER1
  ENTRY POINT:        USER1
  STK LENGTH:       24
  PRIORITY:       35
  DFLT EXCHG:       RQL2EX
  TASK DESCRIPTOR NAME:
  EXTRA:      0
TASK NAME:        USER2
  ENTRY POINT:        USER2
  STK LENGTH:       28
  PRIORITY:       135
  DFLT EXCHG:       USXCH2
  TASK DESCRIPTOR NAME:
  EXTRA:      0
TASK NAME:        USER3
  ENTRY POINT:        USER3
  STK LENGTH:       28
  PRIORITY:       150
  DFLT EXCHG:       USXCH3
  TASK DESCRIPTOR NAME:
  EXTRA:      0
TASK NAME:        ***
:
: ALL USER DEFINED EXCHANGES WHICH ARE TO BE PLACED IN THE INITIAL
: EXCHANGE TABLE MUST BE DEFINED HERE. THESE INCLUDE ANY USER DEFINED
: EXCHANGES WHICH ARE NOT CREATED DYNAMICALLY AT RUN TIME.
:
EXCHANGE:       USXCH1
  SCOPE:      EXTERNAL
  INTERRUPT:      NO
EXCHANGE:       USXCH2
  SCOPE:      EXTERNAL
  INTERRUPT:      NO
EXCHANGE:       USXCH3
  SCOPE     EXTERNAL
  INTERRUPT:      NO
EXCHANGE:       RESPEX
  SCOPE:      EXTERNAL
  INTERRUPT:      NO
EXCHANGE:       RQL2EX
  SCOPE:      PUBLIC
  INTERRUPT:      NO
EXCHANGE:       ***
DFS:    NO
BASIC:      NO
FORTRAN:        NO
NUCLEUS:        :F1:
EXTENSIONS:       :F1:
:
: USRCOD.LIB CONTAINS USER SUPPLIED OBJECT CODE.
:
LINK:       :F1:USRCOD.LIB
LINK:       ***
CODE:       0
DATA:       3800H
```

```
:
:  THIS IS AN EXAMPLE OF USING THE INTERACTIVE CONFIGURATION UTILITY TO
:  CONFIGURE AN APPLICATION SYSTEM USING THE DFS SERVICES.
:  THE KEY CHARACTERISTICS OF THIS APPLICATION ARE:
:  DFS SERVICES USED: OPEN, CLOSE, READ, WRITE, AND SEEK.
:  USER TASKS: TWO, CALLED UTASK1 AND UTASK2.
:  DISK CONFIGURATION:
:       ONE 201 CONTROLLER, WITH TWO DRIVES.
:       TWO 204 CONTROLLERS. THE FIRST IS CONNECTED TO ONE STANDARD SIZE DRIVE,
:               AND THE SECOND IS CONNECTED TO TWO MINI-SIZE DRIVES.
:       ONE 206 CONTROLLER WITH THREE DRIVES.
:
:
CONFIG:       :F1:DFSCON.OBJ
SUBMIT:       :F1:DFSEXG.CSD
CAM:      :F1:CAMMOD.OBJ
LINKOBJ:        :F1:DFSSYS.LNK
LINKMAP:        :F1:DFSEXG.LEM
LOCOBJ:       :F1:DFSSYS.LOC
LOCMAP:       :F1:DFSEXG.LOM
CAMOBJ:       :F1:CAMMOD
CAMMAP:        :F1:DFSEXG.CLM
BOOT LDR:        NO
LOADABLE:        NO
CPU TYPE:       80/20
TERM HNDLR:        NONE
FSM:       NO
DEBUG:       NONE
ANLG HDLR:        NONE
TASK NAME:        UTASK1
  ENTRY POINT:        UTASK1
  STK LENGTH:        64
  PRIORITY:      150
  DFLT EXCHG:        UTSK1X
  TASK DESCRIPTOR NAME:
  EXTRA:      0
TASK NAME:        UTASK2
  ENTRY POINT:        UTASK2
  STK LENGTH:        80
  PRIORITY:      160
  DFLT EXCHG:        UTSK2X
  TASK DESCRIPTOR NAME:
  EXTRA:      0
TASK NAME:         ***
EXCHANGE:       UTSK1X
  SCOPE:      EXTERNAL
  INTERRUPT:        NO
EXCHANGE:       UTSK2X
  SCOPE:      EXTERNAL
  INTERRUPT:        NO
EXCHANGE:       ***
DFS:      YES
  ATTRIB:       NO
  DELETE:       NO
  FORMAT:       NO
  LOAD:     NO
  RENAME:       NO
  OPEN:     YES
  CLOSE:      YES
  READ:     YES
```

```
WRITE:       YES
  DRSSTK:      48
  DRSPRI:      135
SEEK:      YES
DISKIO:       YES
  DIOSTK:       48
  DIOPRI:      129
CNTRLR:      201
  ADDRESS:      88H
  LEVEL:     2
  CNTLR STK:        80
  CNTLR PRI:       33
  TASK DESCRIPTOR NAME:
  DRIVE NAME:        SA
    UNIT:      0
  DRIVE NAME:        SB
    UNIT:      1
  DRIVE NAME:        ***
CNTRLR:      204
  ADDRESS:       70H
  LEVEL:     3
  CNTLR STK:        80
  CNTLR PRI:       49
  TASK DESCRIPTOR NAME:
  DRIVE NAME:        SC
    UNIT:      0
    SIZE:       STANDARD
    TYPE:       SA800
    CHIP SELECT:       0
    INDEX:       10
      STEP:       8
      SETTLE:      10
      LOAD TIME:       9
  DRIVE NAME:        ***
CNTRLR:      204
  ADDRESS:      60H
  LEVEL:     4
  CNTLR STK:        80
  CNTLR PRI:       66
  TASK DESCRIPTOR NAME:
  DRIVE NAME:        M1
    UNIT:      0
    SIZE:      MINI
    TYPE:      SA400
    CHIP SELECT:       0
    INDEX:       10
      STEP:      20
      SETTLE:      10
      LOAD TIME:       10
  DRIVE NAME:        M2
    UNIT:      1
    SIZE:      MINI
    TYPE:      SA400
    CHIP SELECT:       0
    INDEX:       10
      STEP:      20
      SETTLE:      10
      LOAD TIME:       10
  DRIVE NAME:        ***
  CNTRLR:        206
    ADDRESS:        90H
    LEVEL:       5
    CNTLR STK:        80H
    CNTLR PRI:        84
    TASK DESCRIPTOR NAME:
    DRIVE NAME:        H1
      UNIT:        0
    DRIVE NAME:        H2
      UNIT:        1
    DRIVE NAME:        H3
      UNIT:        5
    DRIVE NAME:        ***
  CNTRLR:        ***
  FILES:       4
  BUFFER:        ***
BASIC:        NO
FORTRAN:        NO
NUCLEUS:        :F1:
EXTENSIONS:        :F1:
LINK:        :F2:UTASK1.OBJ
LINK:        :F2:UTASK2.OBJ
LINK:        ***
CODE:        0
DATA:        3800H
CAM ADR:        ?
  THE CAM MODULE ADDRESS PARAMETER CAN ASSUME ANY OF THE FOLLOWING
  VALUES:
  NO, RELOCATABLE, OR ANY NUMBER BETWEEN 0 AND 65,535 INCLUSIVE IS ACCEPTABLE.
  NO— NO CAM MODULE WILL BE CREATED.
  RELOCATABLE— THE CAM MODULE WILL BE LOCATED WITH THE REST OF THE SYSTEM.
  ANY NUMBER— THE CAM MODULE WILL BE LOCATED AT THIS ADDRESS BEFORE BEING
  LINKED WITH THE REST OF THE SYSTEM.
CAM ADR:        76ADH
```